Enhanced D* Lite Algorithm for Autonomous Mobile Robot

Velappa Ganapathy
Department of Electrical Engineering, Faculty of Engineering
University of Malaya, Malaysia
E-mail: velappa.ganapathy@um.edu.my

Soh Chin Yun
School of Engineering, Monash University Sunway Campus
Jalan Lagoon Selatan, 46150 Bandar Sunway, Malaysi
E-mail: mscysoh@gmail.com, Phone : +603-5514 6000

Tee Wee Chien
School of Engineering, Monash University
Sunway Campus, Malaysia
E-mail: wct_1986@yahoo.com

Abstract

Autonomous Mobile Robot (AMR) has been widely used in exploration and navigation especially where static, dynamic and unknown environment is involved. In this research, Enhanced D Lite Algorithm is developed. The research is divided into two stages where the first stage is to develop a MATLAB simulation to verify the feasibility and the validity of the algorithm. The second stage involves developing the proposed Enhanced D* Lite Algorithm, investigating and verifying this Enhanced D* Lite Algorithm in the real time implementation using Team AmigoBot™. The performances of the Enhanced D* Lite Algorithm in both simulation and real time implementation are investigated and evaluated on various aspects. The results obtained from both simulation and real time implementation have proven the robustness and practicality of the Enhanced D* Lite Algorithm.*

Keywords: Autonomous Mobile Robot (AMR), Mobile Robot, D* Algorithm, D* Lite Algorithn, Enhanced D* Lite Algorithm, Team AmigoBot™ and MATLAB.

1. Introduction

The development of technology has brought great conveniences to human beings. Autonomous robotics plays a vital part in accomplishing useful tasks without human intervention in real world and unmodified environments that have not been specifically engineered for the robot (Saffiotti, 1997). A variety field of study is inter-related with robotics. They are such as Control, Vision, Mechanics, Electrical and Electronics (EE), Information Technology (IT) and Artificial Intelligence (AI) where Vision, Control and Artificial Intelligence (AI) are strongly related with mobile robot navigation. The inter-relationship between robotics and other fields of study is shown in Figure 1 (Bräunl, 2003). The successful use of an Autonomous Mobile Robot (AMR) (Hachour, 2008) is strongly related to its controller. Most autonomous robots use sensors to correspond with the environment (Testler, 2009). The controller of these autonomous robots acts according to the information received by the sensor. Control of robotics systems has many challenging problems especially in unstructured environments. The majority of robots operates in structured environments and performs simple repetitive tasks.

However, operations in unstructured environments require robots to perform more complex tasks that may involve autonomous navigation, compliant motion, operational redundancy, fault tolerance, end-effector positioning, and flexing of links and joints. These operations often have to be completed in dynamic, remote, hazardous or, in general, uncertain surroundings. Humans cannot have exact and complete prior knowledge of these environments as many details are usually unknown, the position of people and objects cannot be predicted a priori, passageways may be blocked, and so on. Furthermore, knowledge acquired through sensing is affected by uncertainty and imprecision. The quality of sensor information is influenced by the sensor's noise, the limited field of view, the conditions of observation, and the inherent difficulty of the perceptual interpretation process. Thus, major challenge of autonomous robotics is the large amounts of uncertainty that characterizes real world environments. Current research in robotics aims to build an autonomous and intelligent robot which can plan its motion in a dynamic and uncertain environment. According to Testler (2009), robot is a programmable machine that imitates the action or appearance of an intellectual creature such as a human being or an animal. This electro-mechanical device can perform autonomous or preprogrammed task. These autonomous or preprogrammed tasks can be any of the dangerous jobs, dirty jobs and boring jobs that human don't want to or don't feel like doing.

A number of software has been developed to control actions and movements of mobile robots to accomplish those preprogrammed or autonomous tasks. All these software needs effective motion planning. Motion planning is often designed to optimize specific performance criteria and to satisfy constraints on the robot's motion (Kam *et al.*, 1997). Typical performance criteria are minimum time to arrive at a milestone and minimum control effort. Typical constraints in the motion planning are obstacle avoidance and maximum robot velocity. But apparently, due to the lack of in-depth consideration from all aspects or the misunderstanding which occurs during the requirements capturing processes, some problems were found in these developed systems. More and more problems have arisen in the development of the robots boasting. Among them, the problem of getting the accurate and shortest path in the robot navigation is quite apparent in the development of robotics. The robot navigation can be accomplished through behavior arbitration (Saffiotti, 1997) and hierarchical behavior control (Tunstel *et al.*, 1997). Some of these basic behaviors are Goal Seeking, Obstacle Avoidance and Wall Following.

Mobile robot is one of the developed advanced technologies which assist humans in various purposes and tasks in many application fields such as education, service, industry, exploration, transportation, and more. In exploration and observation field, robot is used to explore the environment that is new or not reachable by humans. ROBOVOLC (Sim *et al.*, 2003) is one of the examples of robots which designed for exploration and measurement in volcanic environment as it might be dangerous for volcanologist to do. In industry application, mobile robot is mainly used for transporting the materials from one point to another. It is known as Automated Guided Vehicle (AGV) (Garcia *et al.*, 2007). By using AGV in the industry, the cost can be minimized and efficiency can be increased. Mobile robot relies on sensors to get information about their surroundings (Testler, 2009). Normally, mobile robot is equipped with various types of sensors in order to guide the robot to achieve its purpose. Some sensor appliances, image capturing devices or a combination of both sensing and image capturing devices can be treated as supporting appliances in robots navigation. These appliances capture image data from these devices and use vision algorithms for analysis of the terrain (Howard *et al.*, 2001) or sensor fusion to translate different sensory inputs into reliable estimates and environmental models (Kam *et al.*, 1997).

These image data are utilized in order to get a better reading of obstacles' features such as the distance between the robot and obstacle, the size of the obstacle and so on. Thus, location and type of sensors are essential aspects in the mobile robot navigation. Choosing the right sensors for the robot and locating them on the robot effectively are important in getting the best performance of the robot. The number of sensors installed on the robot also needs to be considered and this is to be as few as possible. Navigation is one of the important purposes that uses mobile robot (Franz & Mallot, 2000). Mobile robot navigation is the technique to guide the mobile robot moving from the starting position towards the desired goal, along a desired path or accomplishing the desired purposes where static, dynamic, known and unknown environment is involved (Velappa *et al.*, 2009a). The environment of mobile robot navigation is distinguished by variable terrain, a set of distinct objects or certain objects such as obstacles, milestones, and landmarks (Kam *et al.*, 1997) that may block the movement of the robot in reaching the desired destination or accomplishing the desired purposes. The environment for navigation could be known or unknown with moving and static obstacles.

The environment with static obstacles is called static environment meanwhile the environment with moving obstacles that causes the locations of the obstacles change with time is called dynamic environment. Unknown environment is the environment where the locations of the obstacles are unknown to the mobile robot while known environment is the environment where the locations of the obstacles are known to the mobile robot before navigation is carried out. One common requirement for the mobile robot, independent of the operating domain, is the capability of intelligent navigation. Autonomous Mobile Robot (AMR) has to find the near optimal collision free path from the starting point to the desired goal. Some of the mobile robot navigation purposes are such as navigation for exploring unknown planet (Fu *et al.*, 2006; Eustice *et al.*, 2008), navigation for military purpose (Alvarez *et al.*, 2004; Kim & Eustice, 2009), goal directed navigation (Velappa *et al.*, 2009a; Velappa *et al.*, 2009b; Velappa *et al.*, 2009c) and path finding (Wang & Liu, 2004).

For goal-directed navigation, a mobile robot has to move from the given starting position towards the goal position with the capability to avoid obstacles. The mobile robot should also be able to traverse in both known and unknown environments. Besides, time to traverse from starting point to goal point is also an important aspect emphasized by researches. It is important for mobile robot to plan a shortest path from starting point to goal point. Moreover, the mobile robot must be able to re-plan its path quickly if there is a new obstacle in front or nearby. Some of the existing goal-direct navigation algorithms are such as A* Algorithm (Lester, 2009), D* Algorithm (Stentz, 1994; Ferguson & Stentz, 2006; Choset *et al.*, 2007), and D* Lite Algorithm (Koenig & Likhachev, 2002; Likhachev & Koenig, 2002; Koenig & Likhachev, 2005).

A* Algorithm (Lester, 2009) is an early developed popular graph search algorithm which finds the shortest path from a given initial start node to the goal node. A* Algorithm uses distance plus path cost function to determine the shortest path to the goal node. In A* Algorithm, node or square notation is used rather than coordinate because a map is divided into small grids or squares and nodes represent the center point of each grid D* Algorithm, which is also known as Stentz algorithm or Dynamic A* Algorithm, is developed by Anthony Stentz in 1994. It is better than A* Algorithm because it could be used in partially or completely unknown and also dynamic environment. Dynamic environment means there might contain moving obstacles. A* Algorithm is a simple algorithm which could be only used in static environment. But D* Algorithm is able to repair or update the map of the dynamic environment and it can re-plan quickly whenever it detects there is a new obstacle or an obstacle is removed on the way to goal node (Stentz, 1994).

D* Lite Algorithm (Koenig & Likhachev, 2002; Likhachev & Koenig, 2002; Koenig & Likhachev, 2005) is one of most popular goal-directed navigation algorithm that is widely used for mobile robot navigation in unknown environment. D* Lite Algorithm is a reverse or backward searching method and it is able to re-plan from current position when there is a new obstacle blocking the path. It determines the same paths as D* Algorithm and moves the mobile robot the same way but it is algorithmically different from D* Algorithm. D* Lite Algorithm is developed by Koenig and Likhachev (Koenig & Likhachev, 2002; Likhachev & Koenig, 2002; Koenig & Likhachev, 2005) based on Lifelong Planning A* (Koenig *et al.*, 2004) Algorithm in 2002. It has been widely used for mobile robot navigation in unknown environment. Similarly, it divides the environment into grids and path finding and robot's movement are from grid to grid.

From the study and review of existing D* Lite Algorithm, the main problems of the algorithm are mobile robot is traversing across obstacles' sharp corners, traversing in between two obstacles, and trapped in the 'U-Shaped' type obstacles. Also there is no real time implementation reported. To overcome the mentioned problems, Enhanced D* Lite Algorithm has been implemented. In this research, the existing D* Lite Algorithm is investigated in detail and enhancements are emphasized on (i) preventing mobile robot from traversing across obstacles' sharp corners, (ii) avoiding complicated obstacles such as U-shaped obstacles, or obstacles that trap the mobile robot from moving, (iii) preventing mobile robot from traversing in between two obstacles, (iv) creating virtual wall if necessary, and (v) removing unnecessary pathways to yield shortest path. The performances of the Enhanced D* Lite Algorithm in both simulation and real time implementation are investigated and evaluated on various aspects including (i) the ability to avoid obstacles (ii) finding a shortest path from goal to start position in dynamic environment (back propagation method), (iii) remembering the path, and (iv) following back the same path and returning to the goal position.

2. System Overview

Consider a mobile robot navigation task in an environment with 'U-shaped' type obstacles as shown in Figure 2, where no information about the obstacles positions is provided to the mobile robot. It has to find a shortest path from the goal position towards starting position (back propagation). It will compute a shortest path from its current position with respect to the starting position until it reaches starting position. In this research, the mobile robot environment is represented by grids (Michalewicz, 1994). White grids are known to be traversable, black grids are known to be obstacles, which are untraversable and round dot representing mobile robot. Referring to Figure 2, the mobile robot will traverse vertically up towards the starting position since that is the shortest path to reach starting position from goal position and it assumes all grids are traversable. As the mobile robot is traversing vertically towards starting position, it is trapped in the 'U-shaped' type obstacles as shown in Figure 2B. Eventually the sensors of the mobile robot will detect and realize that there is an obstacle blocking in front of the mobile robot.

The mobile robot will continue to search for other grids and check if there is a new path to reach starting position. The only traversable grid now is the grid behind the mobile robot which means the mobile robot has to traverse out from the 'U-shaped' type obstacles. As the mobile robot traverses out from the 'U-shaped' type obstacles, virtual walls will be created to make sure the same grids would not be traversed again. This is shown in Figure 2C and Figure 2D. The mobile robot is able to escape out from the 'U-shaped' type obstacles and continue to find the shortest path in order to reach starting position. Referring to Figure 2D, as the mobile robot traverses across obstacle's sharp corner, a minimum clearance distance is provided between obstacle's sharp corner and mobile robot. This is important to prevent any damages of mobile robot. Finally, the mobile robot is able to reach starting position as shown in Figure 2E. Once the mobile robot has reached the starting position, it will follow back the path found just now and return to goal position as illustrated in Figure 2F and Figure 2G. No more path finding is required. For the shortest path found travelling from goal to starting position, if there are unnecessary pathways, it would be removed as shown in Figure 2H. This is the navigation strategy of Enhanced D* Lite Algorithm.

3. Enhanced D* Lite Algorithm

Basically, the path finding principle of Enhanced D* Lite Algorithm is the same as existing D* Lite Algorithm. However, Enhanced D* Lite Algorithm has been implemented with five additional features which are mentioned in Section 1. Coordinate specification used to represent positions in Enhanced D* Lite Algorithm is (i, j) where *i* is the row and *j* is the column. Enhanced D* Lite Algorithm is using back propagation searching method as in D* Lite Algorithm to find the shortest path. Back propagation searching method is where the path finding search will start from goal position towards starting position. Details of how the Enhanced D* Lite Algorithm works would be explained in the following steps.

(i) Calculate *h* (Heuristic) value

The first step of the Enhanced D* Lite Algorithm is to calculate the *h* (heuristic) value. Starting position is assigned with an *h* value of 0. The *h* value is incremented by 1 from grid to grid until all grids are assigned with the respective *h* values as seen in Figure 3. Increment of *h* value is done vertically, horizontally, and diagonally with the condition that the smallest h value is taken from the starting position to the goal position.

(ii) Calculate *rhs* (Look Ahead Function) and *g* (Cost Function) values

Initially, each grid of the environment is assigned with *g* and *rhs* values to infinity. *rhs(i, j)* is a look ahead function. *rhs* values will be re-calculated when the grids are analyzed. Formula of the *rhs* is shown below.

$$rhs(i, j) = \min[succ(i, j)(g)] + 1 \qquad (1)$$

$$where\ g = g\ values\ of\ rhs(i, j)\ successors$$

g(i, j) is cost function and *g* value will be re-calculated when the grid is expanded. The formula of *g* is shown below.

$$g(i, j) = rhs(i, j) \qquad (2)$$

(iii) Calculate *k1* and *k2* (Priority Queue Function) values

Basically, *k1* and *k2* are priority queue functions. *k1* and *k2* values would be calculated when the grid(s) is/are analyzed by mobile robot. Grid with the smallest *k1* value will be expanded. Formula for calculating *k1* and *k2* are shown below.

$$k1(i, j) = rhs(i, j) + h(i, j) \qquad (3)$$

$$k2(i, j) = \min[g(i, j), rhs(i, j)] \qquad (4)$$

Path finding will stop when *k1* value is equal to *k2* value. This indicates that the mobile robot has reached the starting position.

(iv) Initialization

After defining all the variables and their respective functions in Enhanced D* Lite Algorithm, initialization is to be performed before path finding starts. First, *rhs* value of goal position is set to value of 0, *k1* and *k2* values of goal position are calculated and *g* value of goal position is set equal to *rhs* value of goal position.

(v) Compute Shortest Path

Next, the surrounding grids will be analyzed using sonar sensors. This is to confirm if the grid is free of obstacles and could be traversed. In this research, the mobile robot, Team AmigoBot™ (MobileRobots, 2007), is used. Team AmigoBot™ comes with eight built-in sonar sensors. The structure of the Team AmigoBot™ is shown in Figure 4.

Referring to Figure 4, *s1* to *s6* are the front sensors and *s7* and *s8* are the rear sensors. The angles of all the mounted sensors are shown in Figure 5. Positive angle is to the left and negative angle is to the right. Array *S* is used to store the sonar sensors' readings obtained from the environment. *S* is represented mathematically as below:

$$S = [s1\ s2\ s3\ s4\ s5\ s6\ s7\ s8] \qquad (5)$$

where *s*1-*s*8 are eight sensor readings indicating the configurations of the nearest obstacles to the mobile robot from the current position. Figure 6 shows the position of the mobile robot for grid analysis. Basically, there are eight grids surrounding the mobile robot labeled grid A to grid H. Selection procedures of the grid(s) to be updated with new *rhs*, *k1* and *k2* values are shown in the flowchart in Figure 7. The grid(s) updated with the new values would be the potential next grid to be expanded. Referring to the flowchart in Figure 7, for each of the eight grids surrounding the mobile robot, this Enhanced D* Lite Algorithm will start with the verification if the particular grid is free of obstacle by analyzing the sonar sensors readings. If the grid is not free of obstacle, it will be checked for second condition, whether the grid is already a virtual wall. If the grid is not a virtual wall, it will be checked for third condition, whether the grid is already in the closed list.

If the grid is not in the closed list, it will be checked again for fourth condition, whether grid B is surrounded by obstacles at grid A and grid C, grid D is surrounded by obstacles at grid C and grid E, grid F is surrounded by obstacles at grid E and grid G, and grid H is surrounded by obstacles at grid G and grid A. This condition is used to prevent mobile robot from traversing in between two obstacles. If this condition is not satisfied, then the grid's information will be updated. This updated information is the next step to be travelled. The fourth condition is used to prevent the mobile robot from traversing in between two obstacles. Referring to Figure 8 as an example, although traversing from grid (4, 2) to grid (3, 3) would give shorter path from the goal position to reach starting position, but this action is prevented. The shaded text boxes in the flowchart in Figure 7 are the proposed enhancements.

(vi) Store and update grid information

Referring to the flowchart in Figure 7, if any of the eight grids does not satisfy all the four conditions, the grid would be updated with new *rhs*, *k1* and *k2* values. If any of the four conditions is met, the next grid would be analyzed till all the eight grids are exhausted.

(vii) Priority queue

After analyzing all the eight grids, the updated grid(s) with the smallest *k1* value would be selected where *U* is given by:

$$U = \min(k1) \qquad (6)$$

Then, a grid in the selected list with minimum distance with respect to starting position would be selected and this will be the next grid to be expanded, *(i, j).*

(viii) Traversing to the grid to be expanded

Before mobile robot traversing to the grid to be expanded, few more conditions need to be checked. Firstly, *g(i, j)* value of the grid to be expanded is set to its *rhs(i, j)* value. Secondly, the grid will be checked whether it has been traversed twice. If yes, a virtual wall would be created for the previously traversed grid and the grid to be expanded would be stored in the closed list. This is to make sure that the previously traversed grid would not be traversed again and the grid in the closed list would not be analyzed again. Follow by the verification of next condition. This condition is to determine if the grid to be expanded is a diagonal grid with an obstacle on its left, right, top, or bottom. If the condition is satisfied, the mobile robot will traverse to the grid to be expanded with a minimum clearance distance between mobile robot and obstacles which is shown in Figure 10. Otherwise, the mobile robot will directly traverse to the grid to be expanded. Better illustration of this step is shown in the flowchart in Figure 9. If the grid to be expanded is free of obstacles, list of traversing actions are illustrated in Figure 11 (1 to 6). If the grid to be expanded is a diagonal grid with an obstacle on its left, right, top, or bottom, list of traversing actions are illustrated in Figure 12 (7 to 14).

(ix) Return to goal position

After finding the shortest path, the mobile robot will return to the goal position. Basically, no more searching is required, *PathRemembering* function will be executed. If there are unnecessary pathways, it will be removed while the mobile robot returns. This is illustrated in

Figure **13**. If there are additional obstacles blocking the returning path, step 5 to step 8 will be repeated to find the new shortest path. Thus, the current position will be assigned as new goal position and the original goal position will be assigned as new starting position.

The complete flowchart of Enhanced D* Lite Algorithm back propagation searching method from goal position towards starting position is shown in

Figure **14**. For returning path, mobile robot returns to the goal position by following the shortest path grids found. The movement of the mobile robot would still be from grid to grid for further analysis if there is any new obstacle blocking the shortest path found. Figure 15 illustrates the detailed flowchart of Enhanced D* Lite Algorithm back propagation returning method from the starting position to the goal position. The research is divided into two stages where the first stage is to create and enhance the existing D* Lite Algorithm using MATLAB programming to develop a MATLAB simulation to verify feasibility and validity of the algorithm. The second stage involves developing the Enhanced D* Lite Algorithm by incorporating Team AmigoBot™, investigating and verifying this Enhanced D* Lite Algorithm in the real time implementation using Team AmigoBot™. The details of MATLAB simulation and real time implementation are discussed in Section 4.

4. Results and Discussions

4.1 MATLAB Simulation

Basically, MATLAB simulation is created to confirm the feasibility of the Enhanced D* Lite Algorithm. In MATLAB simulation, each grid is represented as one unit square. Instead of using sonar sensor readings to identify obstacles, the position of the obstacles are either randomly generated or user defined.

Simulation test, named Test-1, is established according to the specification in Table 1. The simulated environment as shown in
Figure **16** is generated from the details shown in Table 1. The goal is located on the right bottommost grid and the starting point is located on the left upmost grid. The simulated mobile robot is expected to find a collision free path from the goal position towards starting position through back propagation searching method. After identifying the collision free path, unnecessary paths are removed. The simulated mobile robot is expected not to travel to grid that falls in the unnecessary pathway and return from the starting position to the goal position in a shortest collision free path.

In Test-1, other than looking into the travelling activity of the simulated mobile robot from goal position to the start position, we have looked into the circumstances where (a) additional obstacles are not included; (b) additional obstacles are included in the returning path from starting position to the goal position.
Figure **17** illustrates the travelling path from goal position to the starting position of Test-1. Returning path from the starting position to the goal position without additional obstacles in Test-1a is shown in Figure 18 while returning path from the starting position to the goal position with additional obstacles in Test-1b is shown in Figure 19.

Result of Test-1: Travelling from goal to start position

Referring to the result of Test-1 as in
Figure **17**, mobile robot had travelled from the goal position to the starting position (back propagation), a minimum clearance distance was provided between obstacles' sharp corners and virtual walls were created so that the same grids would not be traversed or analyzed again. Moreover, mobile robot was able to move away from 'U-shaped' type obstacles and it did not traverse in between two obstacles. Trace line (without considering the trace line covered with virtual walls) is the shortest path found by travelling from goal to starting position.

Result of Test-1a: Returning from starting position to goal position WITHOUT additional obstacles

Referring to Figure 18, it shows the result obtained by Test-1a, the mobile robot returning from starting position to goal position without additional obstacles. For returning path without additional obstacles blocking the initial shortest path found, *PathRemembering* function was executed to direct the mobile robot back to the goal position through grid by grid movement and no more path searching was required. In order to avoid any unexpected additional obstacles, before mobile robot is directed by *PathRemembering* remember function travelling to the next grid, the Enhanced D* Lite Algorithm will identify if the next grid is blocked with additional obstacles. Unnecessary pathways were removed in order to enable the mobile robot returning from goal position to starting position in the shortest collision free path. The returning path from the starting position to the goal position with no additional obstacles is shown in Figure 18.

Result of Test-1b: Returning from starting position to goal position WITH additional obstacles

The result of mobile robot returning from starting position to goal position with additional obstacles as in Test-1b is shown in the Figure 19. While the mobile robot was directed by *PathRemembering* function to the next grid, it detected that there was an obstacle, immediately the current position of the mobile robot became goal position and the original goal position became starting position. Then *ComputeShortestPath* function was executed again to find the shortest path from the new goal position to the new starting position through back propagation searching method. From Test-1, all the five additional features of Enhanced D* Lite Algorithm were successfully implemented in the simulation where Enhanced D* Lite Algorithm is capable in (i) preventing mobile robot from traversing across obstacles' sharp corners, (ii) avoiding complicated obstacles, (iii) preventing mobile robot from traversing in between two obstacles, (iv) creating virtual wall if necessary, and (v) removing unnecessary pathways to yield shortest path. Figure 20 uses the returning path from starting position to goal position with no additional obstacles in Test-1 to further illustrates the five additional features of Enhanced D* Lite Algorithm on top of the existing D* Lite Algorithm. In short, MATLAB simulation has verified and has validated the effectiveness of the Enhanced D* Lite Algorithm.

4.2 Real Time Implementation

The Enhanced D* Lite Algorithm is implemented on real life situation using Team AmigoBot™ (MobileRobots, 2007) and the performance is observed. In real life situation, an environment might consist of obstacles, and the locations of the obstacles are unknown for the mobile robot. Moreover, the obstacles might move around from time to time. This indicates that the sonar sensors of the Team AmigoBot™ play an important role in detecting obstacles. Team AmigoBot™ is manufactured by MOBILEROBOTS Inc. is suitable to be used for this research due to its reasonable size, weight, and TCP/IP wireless control. Moreover, the surrounding body of Team AmigoBot™ is equipped with eight sonar sensors. Team AmigoBot™ is the smallest member of the Pioneer family mobile robot produced by MOBILEROBOTS Inc. Team AmigoBot™

can be operated in one of the three available modes: server, maintenance and standalone modes. Figure 21 shows the top view, side view and bottom view of Team AmigoBot™ (MobileRobots, 2007).

The standard AmigoBot comes with eight range-finding sonars: one on each side, four forward facing and two in the rear for 360-degree sensing coverage. Other than that, the Team AmogoBot™ is equipped with two shaft encoders to track (x, y) and theta position. Besides, the ability to communicate via TCP/IP wireless connection is most suitable in this path planning related project. In real time implementation, the dimensions of the environment are in millimeters (mm), and coordinate specification are in (x,y). The working environment for real time implementation of the Enhanced D* Lite Algorithm is a 2500 mm by 2500 mm area. Within the environment area, there are 10 round-shaped type obstacles randomly arranged. For the comparison purpose, the positions of the 10 round-shaped type obstacles are defined by the user. The actual working environment constructed for the real time implementation is shown in Figure 22(a). The mobile robot is expected to find a collision free path from the goal position towards starting position through back propagation searching method.

After identifying the collision free path, unnecessary path is removed. The mobile robot is expected not to travel to grid that lies in the unnecessary pathway and return from the starting position to the goal position in a shortest collision free path. A representation environment is also created using MATLAB to illustrate the real time path taken by the mobile robot. This representation environment will show the movement of the real mobile robot during path finding in real time and at the same time shortest path trace line will be plotted. The trace line is later compared with the shortest path trace line from the MATLAB simulation. The representation environment is scaled to 100:1 from to the actual environment. The two dimensional top view of the representation environment is shown in Figure 22(b). The starting position is represented by the circle on the top location, goal position is represented by circle on the bottom location, mobile robot is represented by round shaped object and all of the obstacles are represented by solid black 'Round-Shaped' boxes. In this research, the starting position is (1450mm, 2150mm) and the goal position is the coordinate (1450 mm, 250 mm).

The above real time implementation, named Test-2, can be summarized as in Tables 2 and 3. Two tests have been carried out and the comparison of shortest path obtained in real time implementation and MATLAB simulation is presented below. In Test-2, other than looking into the travelling activity of the mobile robot from goal position to the start position, we have looked into the circumstances where additional obstacles are (a) not included and (b) included in the returning path from starting position to the goal position. Real time representation of the returning path from the starting position to the goal position without additional obstacles carried out in Test-2a is shown in Figure 23(a) and the real time representation of the returning path from the starting position to the goal position with additional obstacles done in Test-2b is shown in Figure 24(a).

Result of Test-2a: Returning from starting position to goal position WITHOUT additional obstacles

Referring to the results of Test-2a in Figure 23, the same path pattern was obtained for both real time implementation and MATLAB simulation. No additional obstacles were added for the returning path when the mobile robot returned to the goal position in Test-2a. The trace lines as seen in Figure 23 were the shortest path found from the returning path from the starting position to the goal position with no additional obstacles in both real time implementation and MATLAB simulation.

Result of Test-2b: Returning from starting position to goal position WITH additional obstacles

Referring to the results of Test-2b, the same path pattern was obtained for both real time implementation and MATLAB simulation. For Test-2b, two additional obstacles were added for the returning path to block the initial found shortest path. Thus, when the sonar sensors of the mobile robot detected additional obstacles blocking the returning path, immediately *ComputeShortestPath* function was executed to find the shortest path again from the current position. The current position was set to become new goal position and the original goal position was set to become new starting position (back propagation). Thus, the trace line as seen in Figure 24 above new goal position were the initial shortest path found and the trace lines below new goal position were the new shortest path found from the new goal position to the new starting position (back propagation) in both real time implementation and MATLAB simulation. In overall, real time implementation has confirmed the practicality and robustness of the Enhanced D* Lite Algorithm. Some findings from the real time implementations were that the sensors readings were found to be fluctuating, and sometimes inaccurate and inconsistent. Thus, a 'settling time' was allowed to ensure that the sensors readings were stabilized before being captured and analyzed. After allowing a small settling time, the sonar sensors readings were found to be more consistent when estimating the obstacle distances.

5. Conclusions

In this research, the following features of the proposed Enhanced D* Lite Algorithm have been verified and proven with MATLAB simulation and real time implementation: (i) preventing mobile robot from traversing

across obstacles' sharp corners, (ii) avoiding complicated obstacles, (iii) preventing mobile robot from traversing in between two obstacles, (iv) creating virtual wall if necessary, and

(v) removing unnecessary pathways to yield shortest path. This investigated Enhanced D* Lite Algorithm is flexible to enable the mobile robot to reach start position from an arbitrary goal position. The mobile robot is capable of escaping from complicated obstacles such as U-shaped obstacles, or obstacles that trap the mobile robot from moving. Besides, directly traversing across obstacles' sharp corners and moving in between two obstacles are prohibited, since there might be a chance the mobile robot would collide with obstacles. Virtual walls are created to avoid the same position being traversed over and over again. Unnecessary pathways are removed in order to yield the shortest path from the starting position to the desired goal position. In the event of new obstacle(s) blocking the path from the starting position to the goal position, Enhanced D* Lite Algorithm is capable of re-planning quickly and the mobile robot is able to return to the goal position by finding a shortest path again from its current position to the goal position. Real time implementation of the Enhanced D* Lite Algorithm was investigated in detail. The results obtained from real time implementation are compared with the MATLAB simulation to verify the practicality and robustness of the proposed algorithm. It is proven that the Enhanced D* Lite Algorithm is capable of (i) avoiding obstacles (ii) finding a shortest path from goal to start position in dynamic environment (back propagation method), (iii) remembering the path, and finally (iv) following back the same path and returning to the goal position.

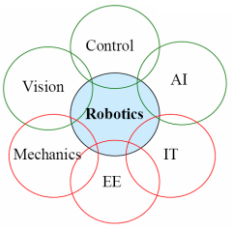

Figure 1: Fields Related to Robotics (Bräunl, 2003)

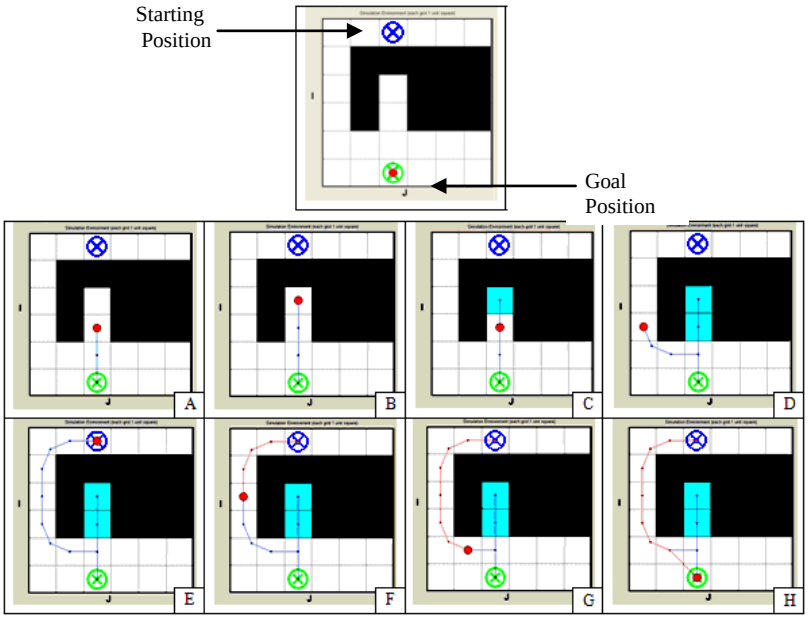

Figure 2: Illustration of the Navigation Strategy

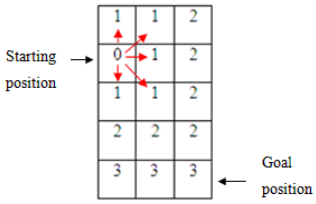

Figure 3: Calculate h (Heuristic) Value

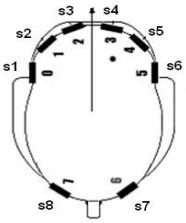

Figure 4: Top View of Team AmigoBot™ with Sensors Labeling (MobileRobots, 2007)

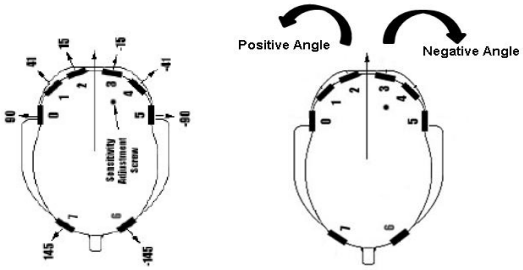

Figure 5: Top View of Team AmigoBot™ with Mounted Angles of Sensors (MobileRobots, 2007)

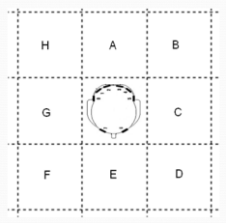

Figure 6: Grids A to H Surrounding the Team AmigoBot™

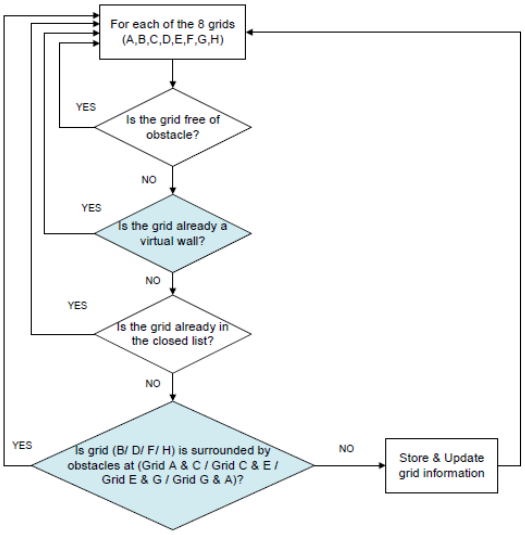

Figure 7: Flowchart Used to Determine which Grid is to be Updated with New *rhs*, *k1* and *k2* Values

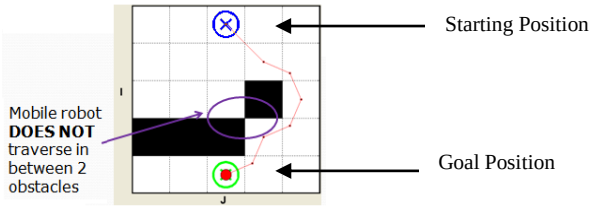

Figure 8: Prevention from Traversing in between Two Obstacles

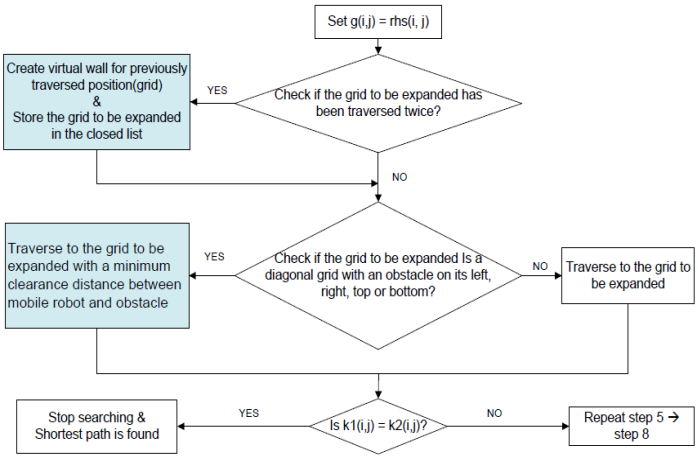

Figure 9: Flowchart Used to Determine which Traversing Action is to be Taken and Virtual Wall(s) is(are) Created when Necessary

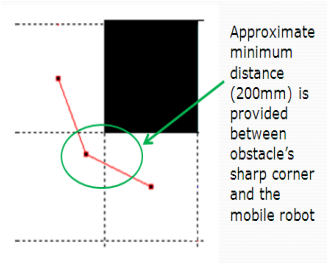

Figure 10: Minimum Clearance Distance is provided between Obstacles' Sharp Corners and Mobile Robot

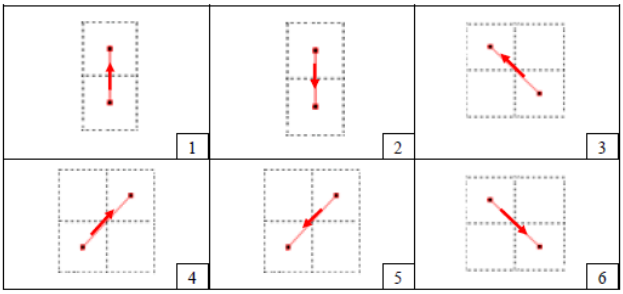

Figure 11: List of Traversing Actions if the Grid to be expanded is Free of Obstacles

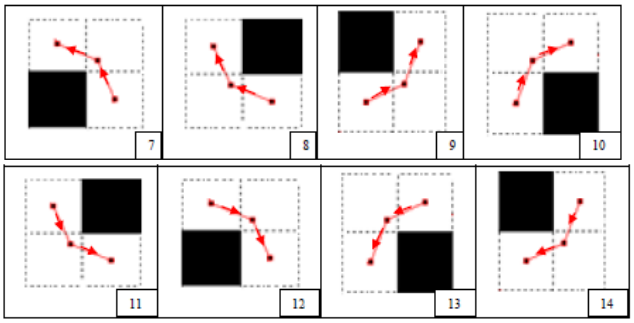

Figure 12: List of Traversing Actions if the Grid to be expanded is a Diagonal Grid with an Obstacle on its Left, Right, Top, or Bottom

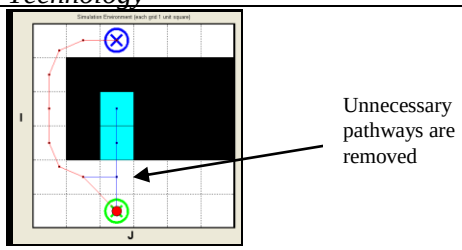

Figure 13: Virtual Walls are Created and Unnecessary Pathways are Removed

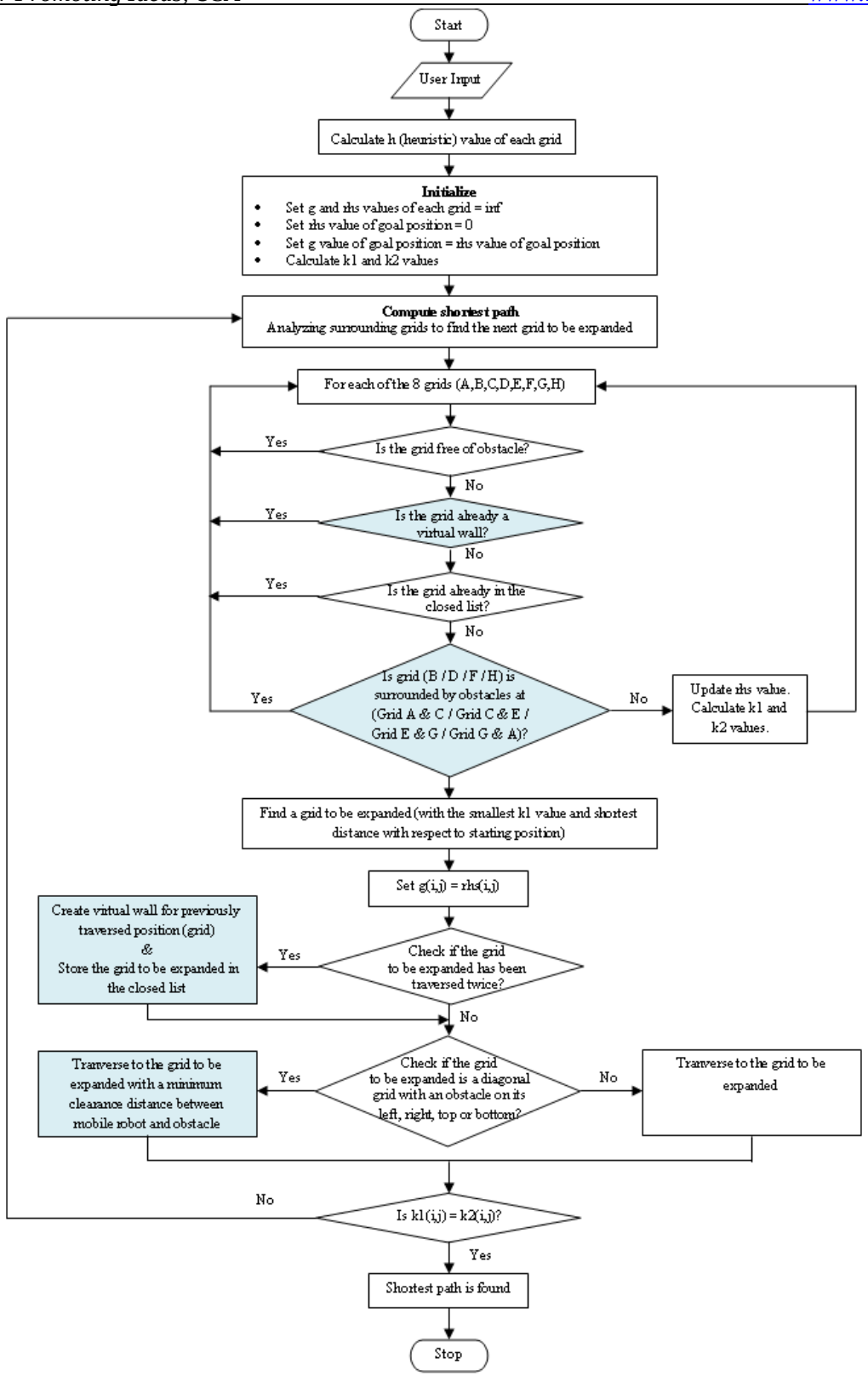

Figure 14: Complete Flowchart of Enhanced D* Lite Algorithm Back Propagation Searching Method from Goal Position Towards Starting Position

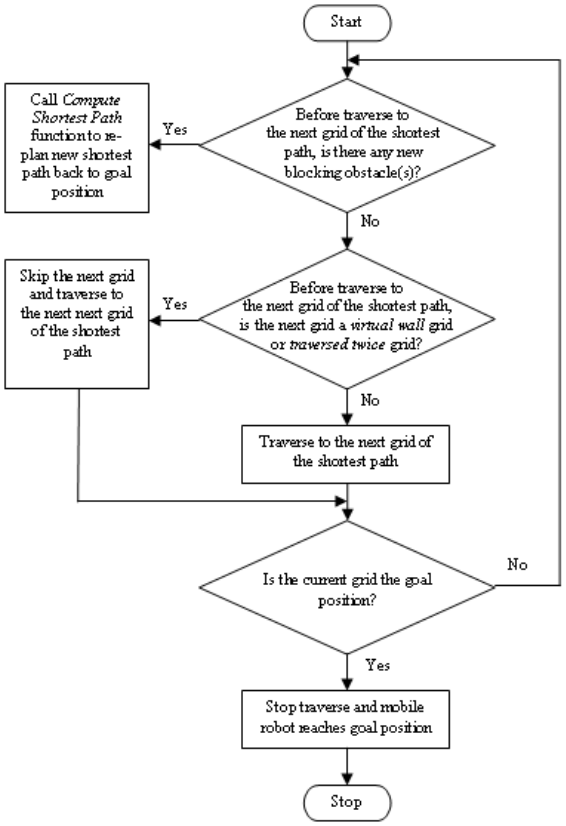

Figure 15: Detailed Flowchart of Enhanced D* Lite Algorithm Back Propagation Returning Method from Starting Position to the Goal Position

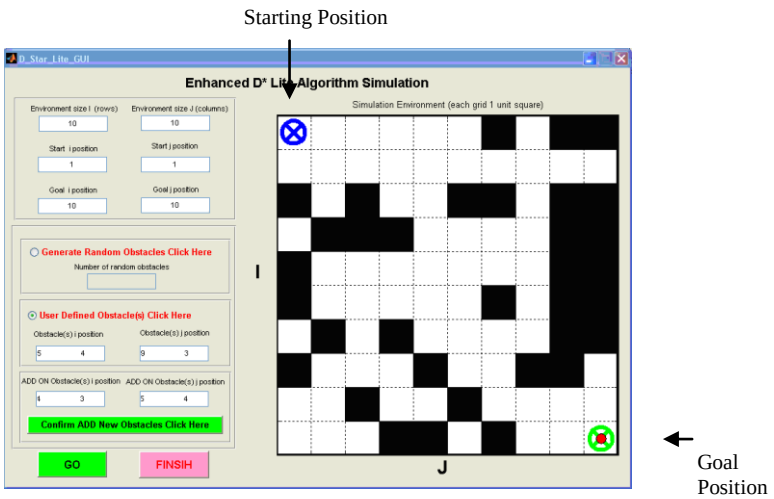

Figure 16: Simulated Environment for Test-1

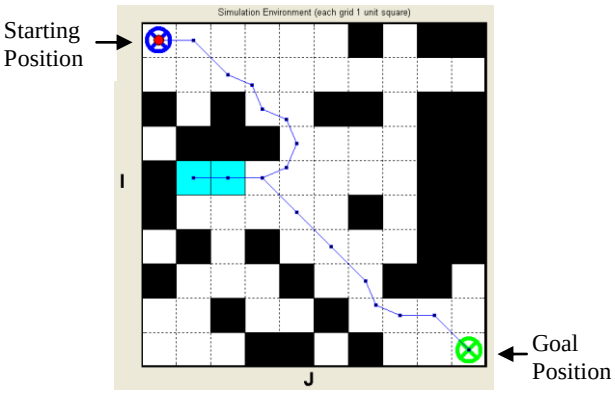

Figure 17: Travelling from Goal to Starting Position

Starting Position

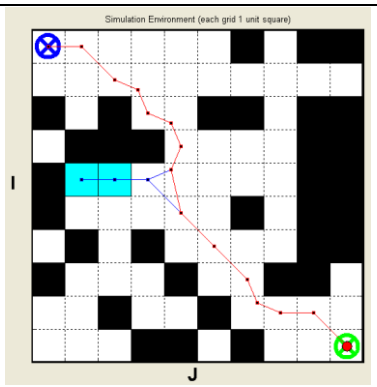

Figure 18: Returning from Starting to Goal Position without Additional Obstacles

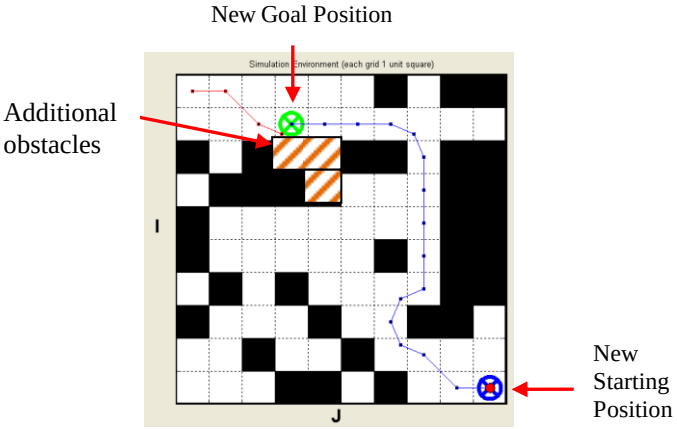

Figure 19: Returning from Starting to Goal Position with Additional Obstacles

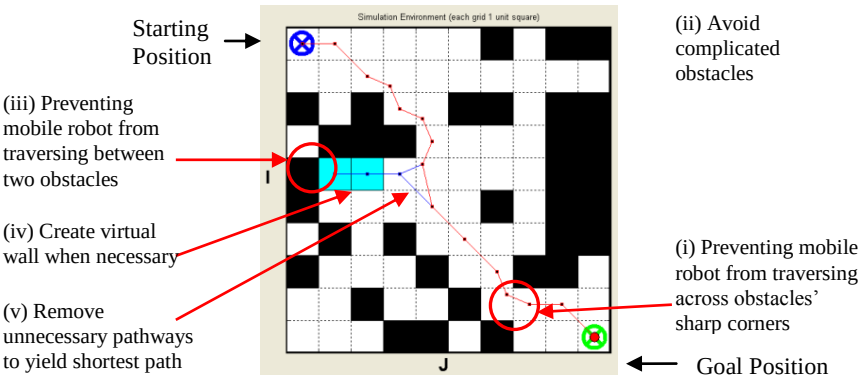

Figure 20: Verification of Five Additional Features of Enhanced D* L g t the Returning Path from Starting Position to Goal Position without Additional Obstacles of Test-1

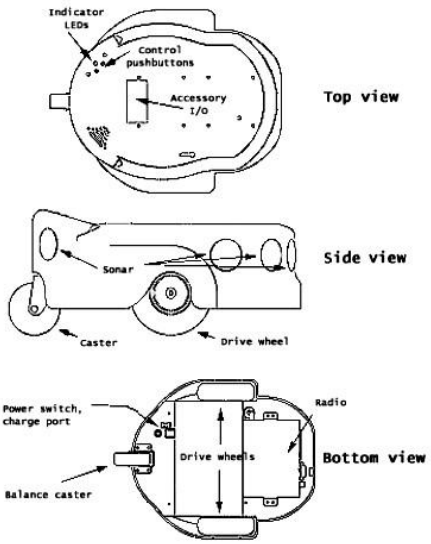

Figure 21: Top, Side and Bottom View of Team AmigoBot™ (MobileRobots, 2007)

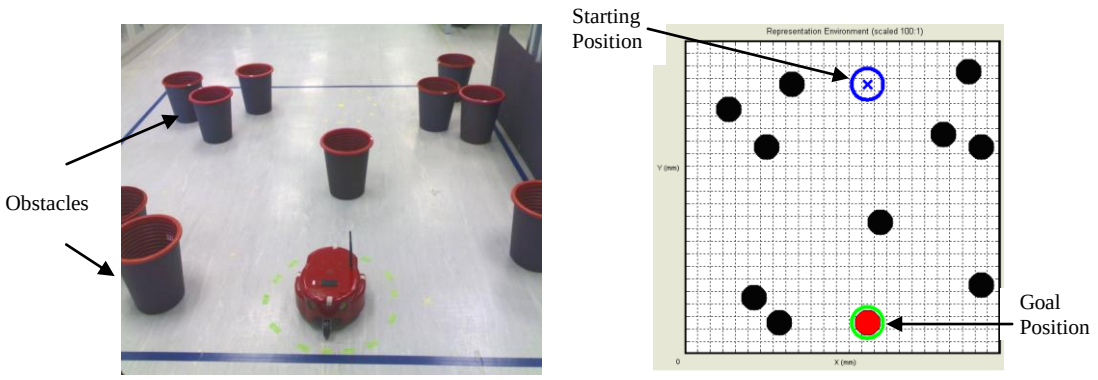

Figure 22: (a) Actual Environment (2500 mm x 2500 mm); (b) Representation Environment Created for Real Time Implementation

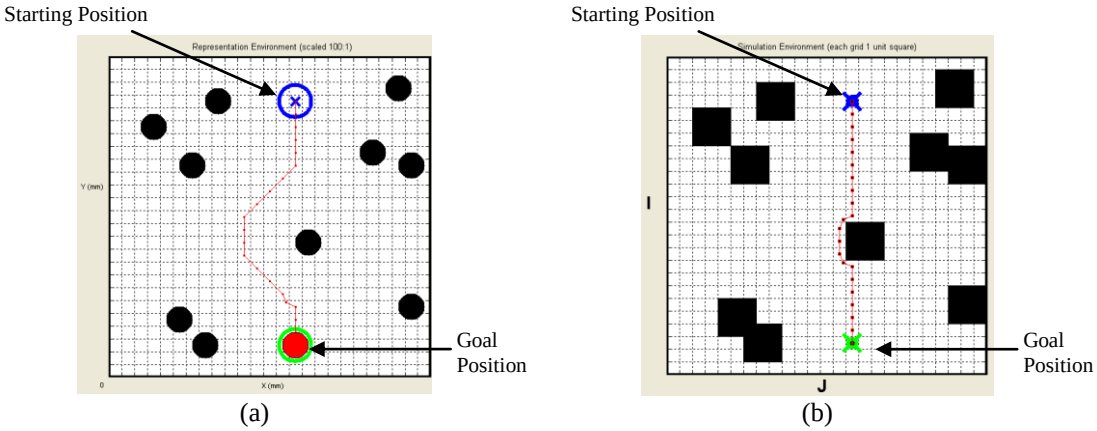

Figure 23: Result of Test-2a (a) Real Time Implementation Representation Environment; (b) MATLAB Simulation Environment

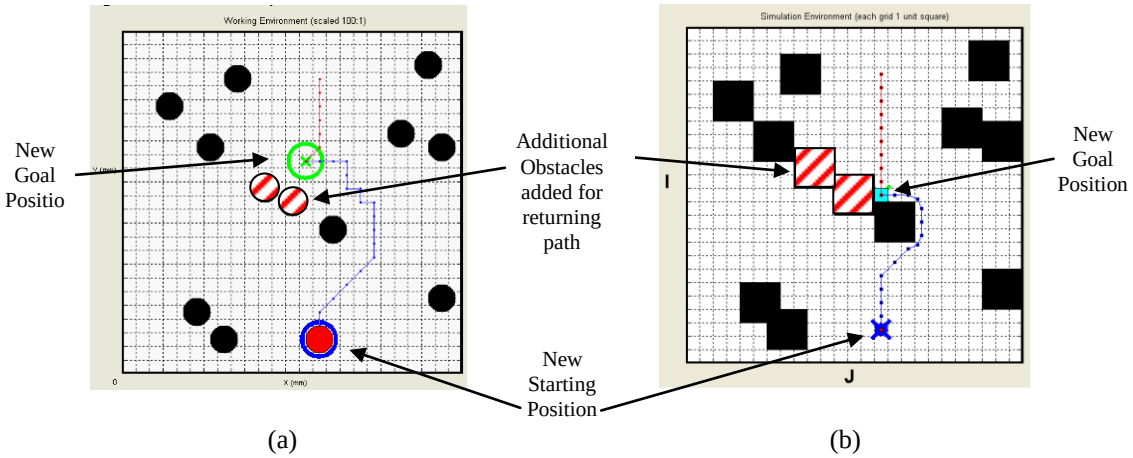

Figure 24: Result of Test-2b (a) Real Time Implementation Representation Environment; (b) MATLAB Simulation Environment

Environment size	10 x 10
Starting position	1,1
Goal position	10,10
Obstacles	34 obstacles positions defined by user
Additional Obstacles added for returning path	Initially no, when Test-1 repeated, additional obstacles for returning path are added.

Table 1: Input Parameters of Test-1

Environment size	2500 x 2500
Starting position	1450,2150
Goal position	1450,250
Obstacles	10. Positions defined by user
Additional Obstacles added for returning path	NO.

Table 2: Input Parameters of Test-2a (Numbers in the Second Column are in mm)

Environment size	2500 x 2500
Starting position	1450,2150
Goal position	1450,250
Obstacles	10. Positions defined by user
Additional Obstacles added for returning path	YES. (1050, 1350) (1250, 1250)

Table 3: Input Parameters of Test-2b (Numbers in the Second Column are in mm)

References

Alvarez, Alberto, Caiti, Andrea, & Onken, Reiner. (2004). "Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean". *IEEE Journal of Oceanic Engineering*, Vol. 29, No. 2, Pages 418 – 429.

Bräunl, Thomas. (2003). *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. Springer, Verlag, Berlin, Heidelberg.

Choset, Howie, Lynch, Kevin, Hutchinson, Seth, Kantor, George, Burgard, Wolfram, Kavraki, Lydia, & Thrun, Sebastian. (2007). *Principles of Robot Motion: Theory, Algorithms, and Implementation*.

Eustice, Ryan M., Pizarro, Oscar, & Singh, Hanumant. (2008). "Visually Augmented Navigation for Autonomous Underwater Vehicles". *IEEE Journal of Oceanic Engineering*, Vol. 33, No. 2, Pages 103 – 122.

Ferguson, Dave, & Stentz, Anthony. (2006). "The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-uniform Cost Environments*". Journal of Field Robotics*, Vol. 23, No. 2, Pages 79 – 101.

Franz, Matthias O., & Mallot, Hanspeter A.. (2000). "Biomimetic Robot Navigation". *Robotics and Autonomous Systems 30*, Pages 131 – 153.

Fu, Yili, Xu, Hongyan, Li, Han, Wang, Shuguo, & Xu, He. (2006). "A Navigation Strategy based on Global Geographical Planning and Local Feature Positioning for Mobile Robot in Large Unknown Environment". *1st International Symposium on Systems and Control in Aerospace and Astronautics (USSCAA)*, Pages 1189 – 1193.

Garcia, Elena, Jimenez, Maria Antonia, Gonzalez de Santos, Pablo, & Armada, Manuel. (2007). "The Evolution of Robotics Research – From Industrial Robotics to Field and Service Robotics". *IEEE Robotics and Automation Magazine*, Volume 14, Issue 1, Pages 90 – 103.

Hachour, O. (2008). "Path Planning of Autonomous Mobile Robot". *International Journal of Systems Applications, Engineering and Development*, Issue 4, Volume 2.

Howard, Ayanna, Tunstel, Edward, Edwards, Dean, & Carlson, Alan. (2001). "Enhancing Fuzzy Robot Navigation Systems by Mimicking Human Visual Perception of Natural Terrain Traversability". *IFSA World Congress and 20th NAFIPS International Conference*, Joint 9th, Volume 1, Pages 7 – 12.

Koenig, Sven, & Likhachev, Maxim. (2002). "Improved Fast Replanning for Robot Navigation in Unknown Terrain". *IEEE International Conference on Robotics and Automation (ICRA '02)*, Volume 1, Pages 968 – 75.

Koenig, Sven, Likhachev, Maxim, & Furcy, David. (2004). "Lifelong Planning A*". *Artificial Intelligence*, Volume 155, Issues 1-2, Pages 93 – 146.

Koenig, Sven, & Likhachev, Maxim. (2005). "Fast Replanning for Navigation in Unknown Terrain". *IEEE Transactions on Robotics*, Vol. 21, No 3, Pages 354 – 363.

Lester, Patrick. (2009). *A* Pathfinding for Beginners.* retrieved 20 May 2009 from <http://www.policyalmanac.org/games/aStarTutorial.htm>.

Likhachev, Maxim, & Koenig, Sven. (2002). "Incremental Replanning for Mapping". *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots ad Systems EPFL*, Switzerland, Volume 1, Pages 667 – 672

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Program*. Second Extended Edition. Spring-Verlag.

MobileRobots Inc.. (2007). *Team AmigoBot™ Operations Manual* version 4. MobileRobots Inc.

Saffiotti, Alessandro. (1997). "Fuzzy Logic in Autonomous Robotics: behavior Coordination". *Proceedings Of the 6th IEEE Int. Conf. on Fuzzy Systems*. Pages 573 – 578.

Sim, Patrick, Sacco, Vincenzo, S. Virk, Gurvinder, & Wang, Xunxian. (2003). "Robot Navigation in Volcanic Environments". *Proceedings of Intelligent Transportation Systems*, Volume. 2, Pages 1010 – 1015.

Stentz, Anthony. (1994). "Optimal and Efficient Path Planning for Partially-Known Environments". *Proceedings IEEE International Conference on Robotics and Automation*, Volume 4, Pages 3310 – 3317.

Testler, Pearl. (2009). *Universal Robots: The History and Workings of Robotics*. The Tech Museum of Innovation. retrieved 18 June 2009 from <http://www.thetech.org/robotics/universal/ index.html>.

Tunstel, Edward, Lippincott, Yanya, & Jamshidi, Mo. (1997). "Behaviour Hirarchy for Autonomous Mobile Robots - Fuzzy-Behaviour Modulation and Evolution". *International Journal of Intelligent Automation and Soft Computing*, Vol. 3 No. 1, Pages 37 – 50.

Velappa, Ganapathy, Soh, Chin Yun, & Ng, Jefry. (2009a). "Fuzzy and Neural Controllers for Acute Obstacle Avoidance in Mobile Robot Navigation". *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2009)*, Suntec Convention and Exhibition Center, Singapore, Pages 1236 – 1241.

Velappa, Ganapathy, Soh, Chin Yun, & Kusuma, Halim Joe. (2009b). "Neural Q-Learning Controller for Mobile Robot". *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2009)*, Suntec Convention and Exhibition Center, Singapore, Pages 863 – 868.

Velappa, Ganapathy, Soh, Chin Yun, & Lui, Wen Lik Dennis. (2009c). "Utilization of Webots and Khepera II as a Platform for Neural Q-Learning Controllers". 2009 *IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009)*, Kuala Lumpur, Malaysia, Pages 783 – 788.

Wang, Meng, & Liu, James N. K.. (2004). "Online Path Searching for Autonomous Navigation". *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics*, Singapore, Pages 746 – 751.